

Best of Two Local Models: Centralized local and Distributed local Algorithms

Guy Even*

Moti Medina*[†]Dana Ron*[‡]

Abstract

We consider two models of computation: centralized local algorithms and local distributed algorithms. Algorithms in one model are adapted to the other model to obtain improved algorithms.

Distributed vertex coloring is employed to design improved centralized local algorithms for: maximal independent set, maximal matching, and an approximation scheme for maximum (weighted) matching over bounded degree graphs. The improvement is three-fold: the algorithms are deterministic, stateless, and the number of probes grows polynomially in $\log^* n$, where n is the number of vertices of the input graph.

The recursive centralized local improvement technique by Nguyen and Onak [NO08] is employed to obtain an improved distributed approximation scheme for maximum (weighted) matching. The improvement is twofold: we reduce the number of rounds from $O(\log n)$ to $O(\log^* n)$ for a wide range of instances and, our algorithms are deterministic rather than randomized.

Keywords. Centralized Local Algorithms, Sublinear Approximation Algorithms, Graph Algorithms, Distributed Local Algorithms, Maximum Matching, Maximum Weighted Matching.

1 Introduction

Local Computation Algorithms, as defined by Rubinfeld et al. [RTVX11], are algorithms that answer queries regarding (global) solutions to computational problems by performing local (sublinear time) computations on the input. The answers to all queries must be consistent with a single solution regardless of the number of possible solutions. To make this notion concrete, consider the *Maximal Independent Set* problem, which we denote by MIS. Given a graph $G = (V, E)$, the local algorithm ALG gives the illusion that it “holds” a specific maximal independent set $I \subseteq V$. Namely, given any vertex v as a query, ALG answers whether v belongs to I even though ALG cannot read all of G , cannot store the entire solution I , and cannot even remember all the answers to previous queries. In order to answer such queries, ALG can probe the graph G by asking about the neighbors of a vertex of its choice.

A local computation algorithm may be randomized, so that the solution according to which it answers queries may depend on its internal coin flips. However, the solution should

*School of Electrical Engineering, Tel-Aviv Univ., Tel-Aviv 69978, Israel.
{guy, medinamo, danar}@eng.tau.ac.il.

[†]M.M was partially funded by the Israeli Ministry of Science and Technology.

[‡]Research supported by the Israel Science Foundation grant number 671/13.

not depend on the sequence of the queries (this property is called query order obliviousness [RTVX11]). We measure the performance of a local computation algorithm by the following criteria: the maximum number of probes it makes to the input per query, the success probability over any sequence of queries, and the maximum space it uses between queries¹. It is desired that both the probe complexity and the space complexity of the algorithm be sublinear in the size of the graph (e.g., $\text{polylog}(|V|)$), and that the success probability be $1 - 1/\text{poly}(|V|)$. It is usually assumed that the maximum degree of the graph is upper-bounded by a constant, but our results are useful also for non-constant upper bounds (see also [RV14]). For a formal definition of local algorithms in the context of graph problems, which is the focus of this work, see Subsection 2.2.

The motivation for designing local computation algorithms is that local computation algorithms capture difficulties with very large inputs. A few examples include: (1) Reading the entire input is too costly if the input is very large. (2) In certain situations one is interested in a very small part of a complete solution. (3) Consider a setting in which different uncoordinated servers need to answer queries about a very large input stored in the cloud. The servers do not communicate with each other, do not store answers to previous queries, and want to minimize their accesses to the input. Furthermore, the servers answer the queries consistently.

Local computation algorithms have been designed for various graph (and hypergraph) problems, including the abovementioned MIS [RTVX11, ARVX12], hypergraph coloring [RTVX11, ARVX12], maximal matching [MRVX12] and (approximate) maximum matching [MV13]. Local computation algorithms also appear implicitly in works on sublinear approximation algorithms for various graph parameters, such as the size of a minimum vertex cover [PR07, NO08, YYI12, ORRR12]. Some of these implicit results are very efficient in terms of their probe complexity (in particular, it depends on the maximum degree and not on $|V|$) but do not give the desired $1 - 1/\text{poly}(|V|)$ success probability. We compare our results to both the explicit and implicit relevant known results.

As can be gleaned from the definition in [RTVX11], local computation algorithms are closely related to *Local Distributed Algorithms*. We discuss the similarities and differences in more detail in Subsection 1.1. In this work, we exploit this relation in two ways. First, we use techniques from the study of local distributed algorithms to obtain better local computation algorithms. Second, we apply techniques from the study of local computation algorithms (more precisely, local computation algorithms that are implicit within sublinear approximation algorithms) to obtain a new result in distributed computing.

In what follows we denote the aforementioned local computation model by CENTLOCAL (where the “CENT” stands for “centralized”) and the distributed (local) model by DISTLOCAL (for a formal definition of the latter, see Subsection 2.3). We denote the number of vertices in the input graph by n and the maximum degree by Δ .

1.1 On the relation between CENTLOCAL and DISTLOCAL

The CENTLOCAL model is centralized in the sense that there is a single central algorithm that is provided access to the whole graph. This is as opposed to the DISTLOCAL model in which each processor resides in a graph vertex v and can obtain information only about the neighborhood of v . Another important difference is in the main complexity measure. In the CENTLOCAL model, one counts the number of probes that the algorithm performs per query,

¹In the RAM model, the running time per query of our algorithms is at most $\text{poly}(ppq) \cdot \log \log n$, where ppq is the maximum number of probes per query and $n = |V|$.

while in the DISTLOCAL model, the number of rounds of communication is counted. This implies that a DISTLOCAL algorithm obtains information about a ball centered at a vertex, where the radius of the ball is the number of rounds of communication. On the other hand, in the case of a CENTLOCAL algorithm, it might choose to obtain information about different types of neighborhoods so as to save in the number of probes. Indeed (similarly to what was observed in the context of sublinear approximation algorithms [PR07]), given a DISTLOCAL algorithm for a particular problem with round complexity r , we directly obtain a CENTLOCAL algorithm whose probe complexity is $O(\Delta^r)$ where Δ is the maximum degree in the graph. However, we might be able to obtain lower probe complexity if we do not apply such a black-box reduction. In the other direction, CENTLOCAL algorithms with certain properties, can be transformed into DISTLOCAL algorithms (e.g., a deterministic CENTLOCAL algorithm in which probes are confined to an r -neighborhood of the query).

1.2 The Ranking Technique

The starting point for our results in the CENTLOCAL model is the *ranking* technique [NO08, YYI12, ARVX12, MRVX12, MV13]. To exemplify this, consider, once again, the MIS problem. A very simple (global “greedy”) algorithm for this problem works by selecting an arbitrary ranking of the vertices and initializing I to be empty. The algorithm then considers the vertices one after the other according to their ranks and adds a vertex to I if and only if it does not neighbor any vertex already in I . Such an algorithm can be “localized” as follows. For a fixed ranking of the vertices (say, according to their IDs), given a query on a vertex v , the local algorithm performs a *restricted* DFS starting from v . The restriction is that the search continues only on paths with monotonically decreasing ranks. The local algorithm then simulates the global one on the subgraph induced by this restricted DFS.

The main problem with the above local algorithm is that the number of probes it performs when running the DFS may be very large. Indeed, for some rankings (and queried vertices), the number of probes is linear in n . In order to circumvent this problem, *random* rankings were studied [NO08]. This brings up two questions, which were studied in previous works, both for the MIS algorithm described above and for other ranking-based algorithms [NO08, YYI12, ARVX12, MRVX12, MV13]. The first is to bound the number of probes needed to answer a query with high probability. The second is how to efficiently store a random ranking between queries.

1.3 Our Contributions

In this section we overview the techniques we use and the results we obtained based on these techniques. See the tables in Section 1.4 for a precise statement of the results.

Orientations with bounded reachability. Our first conceptual contribution is a simple but very useful observation. Rather than considering vertex rankings, we suggest to consider *acyclic orientations* of the edges in the graph. Such orientations induce partial orders over the vertices, and partial orders suffice for our purposes. The probe complexity induced by a given orientation translates into a combinatorial measure, which we refer to as the *reachability* of the orientation. Reachability of an acyclic orientation is the maximum number of vertices that can be reached from any start vertex by directed paths (induced by the orientation). This

leads us to the quest for a CENTLOCAL algorithm that computes an orientation with bounded reachability.

Orientations and colorings. Our second conceptual contribution is that an orientation algorithm with bounded reachability can be based on a CENTLOCAL *coloring* algorithm. Indeed, every vertex-coloring with k colors induces an orientation with reachability $O(\Delta^k)$. Towards this end, we design a CENTLOCAL coloring algorithm that applies techniques from DISTLOCAL colorings algorithms [CV86, GPS88, Lin92, PS10]. Our CENTLOCAL algorithm is deterministic, does not use any space between queries, performs $O(\Delta \cdot \log^* n + \Delta^3)$ probes per query, and computes a coloring with $O(\Delta^2)$ colors. (We refer to the problem of coloring a graph by c colors as c -COLOR.) Our coloring algorithm yields an orientation whose reachability is $\Delta^{O(\Delta^2)}$. For constant degree graphs, this implies $O(\log^* n)$ probes to obtain an orientation with constant reachability. As an application of this orientation algorithm, we also design a CENTLOCAL algorithm for $(\Delta + 1)$ -coloring.

Centralized local simulations of sequential algorithms. We apply a general transformation (similarly to what was shown in [ARVX12]) from global algorithms with certain properties to local algorithms. The transformation is based on our CENTLOCAL orientation with bounded reachability algorithm. As a result we get deterministic CENTLOCAL algorithms for MIS and maximal matching (MM), which significantly improve over previous work [RTVX11, ARVX12, MRVX12], and the first CENTLOCAL algorithm for coloring with $(\Delta + 1)$ colors. Compared to previous work, for MIS and MM the dependence on n in the probe complexity is reduced from $\text{polylog}(n)$ to $\log^*(n)$ and the space needed to store the state between queries is reduced from $\text{polylog}(n)$ to zero.

Deterministic CENTLOCAL-algorithms for approximate maximum matching. We present $(1 - \varepsilon)$ -approximation CENTLOCAL-algorithms for maximum cardinality matching (MCM) and maximum weighted matching (MWM). Similarly to previous related work [NO08, LPSP08, MV13], our algorithm for MCM is based on the augmenting paths framework of Hopcroft and Karp [HK73]. Our starting point is a global/abstract algorithm that works iteratively, where in each iteration it constructs a new matching (starting from the empty matching). Each new matching is constructed based on a maximal set of vertex disjoint paths that are augmenting paths with respect to the previous matching. Such a maximal set is a maximal independent set (MIS) in the intersection graph over the augmenting paths. The question is how to simulate this global algorithm in a local/distributed fashion, and in particular, how to compute the maximal independent sets over the intersection graphs.

By using our CENTLOCAL MIS algorithm (over the intersection graphs), for the case of an approximate MCM, we reduce the dependence of the probe-complexity on n from $\text{polylog}(n)$ [MV13] to $\text{poly}(\log^*(n))$. The space needed to store the state between queries is reduced from $\text{polylog}(n)$ to 0. For the approximate MWM algorithm we also build on the parallel approximation algorithm of Hougardy and Vinkemeir [HV06].

Deterministic DISTLOCAL-algorithms for approximate maximum matching. We present $(1 - \varepsilon)$ -approximation DISTLOCAL-algorithms for MCM and MWM. These algorithms are based on a distributed simulation of the corresponding CENTLOCAL-algorithms. For MCM, we present a deterministic distributed $(1 - \varepsilon)$ -approximation algorithm. The number of rounds

used by the algorithm is

$$\Delta^{O(1/\varepsilon)} + O\left(\frac{1}{\varepsilon^2}\right) \cdot \log^*(n).$$

For MWM, we assume that edge weights are normalized as follows: the maximum edge weight is 1 and w_{\min} denotes the minimum edge weight. We present a deterministic distributed $(1 - \varepsilon)$ -approximation algorithm. The number of rounds used by the algorithm is

$$O\left(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\varepsilon}\right) \cdot \log^* n + \Delta^{O(1/\varepsilon)} \cdot \log(\min\{1/w_{\min}, n/\varepsilon\}).$$

We briefly compare these results with previous results. The best previous algorithms for both the unweighted and weighted cases are by Lotker, Patt-Shamir, and Pettie [LPSP08]. For the unweighted case they give a randomized $(1 - \varepsilon)$ -approximation algorithm that runs in $O((\log(n))/\varepsilon^3)$ rounds with high probability² (w.h.p). Hence we get an improved result when $\Delta^{O(1/\varepsilon)} = o(\log(n))$. In particular, for constant Δ and ε , the number of rounds is $O(\log^*(n))$. Note that an $O(1)$ -approximation of a maximum matching in an n -node ring cannot be computed by any deterministic distributed algorithm in $o(\log^*(n))$ rounds [CHW08, LW08]. For the weighted case, they give a randomized $(1/2 - \epsilon)$ -approximation algorithm that runs in $O(\log(\varepsilon^{-1}) \cdot \log(n))$ rounds (w.h.p).³ Our MWM approximation algorithm runs in significantly fewer rounds for various settings of the parameters Δ , $1/\varepsilon$, and $1/w_{\min}$. In particular, when they are constants, the number of rounds is $O(\log^*(n))$.

1.4 Detailed Comparison with Previous Work

Comparison to previous (explicit) CENTLOCAL algorithms. A comparison of our results with previous CENTLOCAL algorithms is summarized in Table 1. The results assume that Δ and ε are constant. (The dependence of the number of probes and space on Δ and ε is not explicit in [MRVX12, MV13]. For MIS explicit dependencies appear in [ARVX12]. In recent work, Levi et al. [LRY14] show how the exponential dependence on Δ can be reduced to quasi-polynomial in the case of (exact) MIS and MM.) Explicit dependencies on Δ and ε in our result appear in the formal statements within the paper.

Comparison to previous CENTLOCAL oracles in sublinear approximation algorithms. A sublinear approximation algorithm for a certain graph parameter (e.g., the size of a minimum vertex cover) is given probe access to the input graph and is required to output an approximation of the graph parameter with high constant success probability. Many such algorithms work by designing an *oracle* that answers queries (e.g., a query can ask: does a given vertex belong to a fixed small vertex cover?). The sublinear approximation algorithm estimates the graph parameter by performing (a small number of) queries to the oracle. The oracles are essentially CENTLOCAL algorithms but they tend to have constant error probability. Furthermore, the question of bounded space needed to store the state between queries was not an issue in the design of these oracles, since only few queries are performed by the sublinear approximation algorithm. Hence, they are not usually considered to be “bona fide” CENTLOCAL algorithms. A comparison of our results and these oracles appears in Table 2.

²We say that an event occurs with high probability if it occurs with probability at least $1 - \frac{1}{\text{poly}(n)}$.

³Lotker, Patt-Shamir and Pettie remark [LPSP08, Sec. 4] that a $(1 - \varepsilon)$ -MWM can be obtained in $O(\varepsilon^{-4} \log^2 n)$ rounds (using messages of linear size), by adapting the algorithm of Hougardy and Vinkemir [HV06].

Problem	Previous work			Here (Deterministic, 0-Space)
	Space	# Probes	success prob.	# Probes
MIS	$O(\log^2 n)$	$O(\log^3 n)$	$1 - \frac{1}{n}$ [ARVX12]	$O(\log^* n)$ [Coro. 11]
MM	$O(\log^3 n)$	$O(\log^3 n)$	$1 - \frac{1}{n}$ [MRVX12]	$O(\log^* n)$ [Coro. 11]
Δ^2 -COLOR	—	—	—	$O(\log^* n)$ [Thm. 7]
$(\Delta + 1)$ -COLOR	—	—	—	$O(\log^* n)$ [Coro. 11]
$(1 - \varepsilon)$ -MCM	$O(\log^3 n)$	$O(\log^4 n)$	$1 - \frac{1}{n^2}$ [MV13]	$(\log^* n)^{O(1)}$ [Thm. 17]
$(1 - \varepsilon)$ -MWM	—	—	—	$(\Gamma)^{O(1)} \cdot (\log^* n)^{O(1)}$ [Thm. 24]

Table 1: A comparison between CENTLOCAL algorithms. MIS denotes maximal independent set, MM denotes maximal matching, MCM denotes maximum cardinality matching, and MWM denotes maximum weighted matching. Our algorithms are deterministic and stateless (i.e., the space needed to store the state between queries is zero). Since the dependence on Δ and ε is not explicit in [MRVX12, MV13], all the results are presented under the assumption that $\Delta = O(1)$ and $\varepsilon = O(1)$. For weighted graphs, the ratio between the maximum to minimum edge weight is denoted by Γ (we may assume that $\Gamma \leq n/\varepsilon$). The $(1 - \varepsilon)$ -MWM CENTLOCAL-algorithm is of interest (even for $\Gamma = n/\varepsilon$) because it serves as a basis for the $(1 - \varepsilon)$ -MWM DISTLOCAL-algorithm.

In the recent result of Levi et al. [LRY14] it is shown how, based on the sublinear approximation algorithms of Yoshida et al. [YYI12] (referenced in the table), it is possible to reduce the dependence on the failure probability, δ , from inverse polynomial to inverse poly-logarithmic. In particular, they obtain a $(1 - \varepsilon)$ -approximation CENTLOCAL algorithm for MCM that performs $\Delta^{O(1/\varepsilon^2)} \cdot \text{poly}(\log n)$ probes, uses space of the same order, and succeeds with probability $1 - 1/\text{poly}(n)$.

Problem	Previous work			Here	
	# Probes	success prob.	apx. ratio	# Probes	apx. ratio
MIS	$O(\Delta^4) \cdot \text{poly}(\frac{1}{\delta}, \frac{1}{\varepsilon})$	$1 - \delta$	$1 - \varepsilon$ [YYI12]	$\Delta^{O(\Delta^2)} \cdot \log^* n$	1
MM	$O(\Delta^4) \cdot \text{poly}(\frac{1}{\delta}, \frac{1}{\varepsilon})$	$1 - \delta$	$1 - \varepsilon$ [YYI12]	$\Delta^{O(\Delta^2)} \cdot \log^* n$	1
MCM	$\Delta^{O(1/\varepsilon)} \cdot \text{poly}(\frac{1}{\delta}, \frac{1}{\varepsilon})$	$1 - \delta$	$1 - \varepsilon$ [YYI12]	$(\log^* n)^{O(1/\varepsilon)} \cdot 2^{O(\Delta^{1/\varepsilon})}$	$1 - \varepsilon$

Table 2: A comparison between CENTLOCAL oracles in sub-linear approximation algorithms and our CENTLOCAL (deterministic) algorithms. The former algorithms were designed to work with constant success probability and a bound was given on their expected probe complexity. When presenting them as CENTLOCAL algorithms we introduce a failure probability parameter, δ , and bound their probe complexity in terms of δ . Furthermore, the approximation ratios of the sublinear approximation algorithms were stated in additive terms, and we translate the results so as to get a multiplicative approximation.

Comparison to previous DISTLOCAL algorithms for MCM and MWM. We compare our results to previous ones in Table 3. The first line refers to the aforementioned algorithm by Lotker, Patt-Shamir, and Pettie [LPSP08] for the unweighted case. The second line in Table 3 refers to an algorithm of Nguyen and Onak [NO08]. As they observe, their algorithm for approximating the size of a maximum matching in sublinear time can be transformed into a randomized distributed algorithm that succeeds with constant probability (say, $2/3$) and runs in $\Delta^{O(1/\varepsilon)}$ rounds. The third line refers to the aforementioned algorithm by Lotker, Patt-Shamir, and Pettie [LPSP08] for the weighted case. The fourth line refers to the algorithm by Panconesi

and Sozio [PS10] for weighted matching. They devise a deterministic distributed $(1/6 - \varepsilon)$ -approximation algorithm that runs in $O\left(\frac{\log^4(n)}{\varepsilon} \cdot \log(\Gamma)\right)$ rounds, where Γ is the ratio between the maximum to minimum edge weight.

We remark that the randomized CENTLOCAL-algorithm by Mansour and Vardi [MV13] for $(1 - \varepsilon)$ -approximate maximum cardinality matching in bounded-degree graphs can be transformed into a randomized DISTLOCAL-algorithm for $(1 - \varepsilon)$ -approximate maximum cardinality matching (whose success probability is $1 - 1/\text{poly}(n)$). Their focus is on bounding the number of probes, which they show is polylogarithmic in n for constant Δ and ε . To the best of our understanding, an analysis of the probe-radius of their algorithm will not imply a DISTLOCAL-algorithm that runs in fewer rounds than the algorithm of Lotker, Patt-Shamir, and Pettie [LPSP08].

Previous work				Here (Deterministic)	
problem	# rounds	success prob.	apx. ratio.	# rounds	apx. ratio.
MCM	$O(\frac{\log(n)}{\varepsilon^3})$	$1 - \frac{1}{\text{poly}(n)}$	$1 - \varepsilon$ [LPSP08]	$\Delta^{O(\frac{1}{\varepsilon})} + O\left(\frac{1}{\varepsilon^2}\right) \cdot \log^*(n)$	$1 - \varepsilon$ [Thm. 18]
	$\Delta^{O(\frac{1}{\varepsilon})}$	$1 - \Theta(1)$	$1 - \varepsilon$ [NO08]		
MWM	$O(\log(\varepsilon^{-1}) \cdot \log(n))$	$1 - \frac{1}{\text{poly}(n)}$	$1/2 - \varepsilon$ [LPSP08]	$O\left(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\varepsilon}\right) \cdot \log^* n + \Delta^{O(1/\varepsilon)} \cdot \log(\Gamma)$	$1 - \varepsilon$ [Thm. 25]
	$O\left(\frac{\log^4(n)}{\varepsilon} \cdot \log(\Gamma)\right)$	deterministic	$1/6 - \varepsilon$ [PS10]		

Table 3: A comparison between MCM and MWM DISTLOCAL algorithms. The ratio between the maximum to minimum edge weight is denoted by Γ (we may assume that $\Gamma \leq n/\varepsilon$).

2 Preliminaries

2.1 Notations

Let $G = (V, E)$ denote an undirected graph, and $n(G)$ denote the number of vertices in V . We denote the degree of v by $\deg_G(v)$. Let $\Delta(G)$ denote the maximum degree, i.e., $\Delta(G) \triangleq \max_{v \in V} \{\deg_G(v)\}$. Let $\Gamma(v)$ denote the set of neighbors of $v \in V$. The length of a path equals the number of edges along the path. We denote the length of a path p by $|p|$. For $u, v \in V$ let $\text{dist}_G(u, v)$ denote the length of the shortest path between u and v in the graph G . The ball of radius r centered at v in the graph G is defined by

$$B_r^G(v) \triangleq \{u \in V \mid \text{dist}_G(v, u) \leq r\}.$$

If the graph G is clear from the context, we may drop it from the notation, e.g., we simply write $n, m, \deg(v)$, or Δ .

For $k \in \mathbb{N}^+$ and $n > 0$, let $\log^{(k)}(n)$ denote the k th iterated logarithm of n . Note that $\log^{(0)}(n) \triangleq n$ and if $\log^{(i)}(n) = 0$, we define $\log^{(j)}(n) = 0$, for every $j > i$. For $n \geq 1$, define $\log^*(n) \triangleq \min\{i : \log^{(i)}(n) \leq 1\}$.

A subset $I \subseteq V$ is an *independent set* if no two vertices in I are an edge in E . An independent set I is *maximal* if $I \cup \{v\}$ is not an independent set for every $v \in V \setminus I$. We use MIS as an abbreviation of a maximal independent set.

A subset $M \subseteq E$ is a matching if no two edges in M share an endpoint. Let M^* denote a maximum cardinality matching of G . We say that a matching M is a $(1 - \varepsilon)$ -approximate maximum matching if

$$|M| \geq (1 - \varepsilon) \cdot |M^*|.$$

Let $w(e)$ denote the weight of an edge $e \in E$. The weight of a subset $F \subseteq E$ is $\sum_{e \in F} w(e)$ and is denoted by $w(F)$. Let M_w^* denote a maximum weight matching of G . A matching M is a $(1 - \varepsilon)$ -approximate maximum weight matching if $w(M) \geq (1 - \varepsilon) \cdot w(M_w^*)$. We abbreviate the terms maximum cardinality matching and maximum weight matching by MCM and MWM, respectively.

2.2 The CENTLOCAL Model

The model of centralized local computations was defined in [RTVX11]. In this section we describe this model for problems over labeled graphs.

Labeled graphs. An undirected graph $G = (V, E)$ is labeled if: (1) Vertex names are distinct and each have description of at most $O(\log n)$ bits. For simplicity, assume that the vertex names are in $\{1, \dots, n\}$. We denote the vertex whose name is i by v_i . (2) Each vertex v holds a list of $\deg(v)$ pointers, called *ports*, that point to the neighbors of v . The assignment of ports to neighbors is arbitrary and fixed.

Problems over labeled graphs. Let Π denote a computational problem over labeled graphs (e.g., maximum matching, maximal independent set, vertex coloring). A solution for problem Π over a labeled graph G is a function, the domain and range of which depend on Π and G . For example: (1) In the Maximal Matching problem, a solution is an indicator function $M : E \rightarrow \{0, 1\}$ of a maximal matching in G . (2) In the problem of coloring the vertices of a graph by $(\Delta + 1)$ colors, a solution is a coloring $c : V \rightarrow \{1, \dots, \Delta + 1\}$. Let $\text{sol}(G, \Pi)$ denote the set of solutions of problem Π over the labeled graph G .

Probes. In the CENTLOCAL model, access to the labeled graph is limited to probes. A *probe* is a pair (v, i) that asks “who is the i th neighbor of v ?”. The answer to a probe (v, i) is as follows. (1) If $\deg(v) < i$, then the answer is “null”. (2) If $\deg(v) \geq i$, then the answer is the (ID of) vertex u that is pointed to by the i th port of v . For simplicity, we assume that the answer also contains the port number j such that v is the j th neighbor of u . (This assumption reduces the number of probes by at most a factor of Δ .)

Online Property of CENTLOCAL-algorithms. The input of an algorithm ALG for a problem Π over labeled graphs in the CENTLOCAL model consists of three parts: (1) access to a labeled graph G via probes, (2) the number of vertices n and the maximum degree Δ of the graph G , and (3) a sequence $\{q_i\}_{i=1}^N$ of queries. Each query q_i is a request for an evaluation of $f(q_i)$ where $f \in \text{sol}(G, \Pi)$. Let y_i denote the output of ALG to query q_i . We view algorithm ALG as an online algorithm because it must output y_i without any knowledge of subsequent queries.

Consistency. We say that ALG is *consistent with* (G, Π) if

$$\exists f \in \text{sol}(G, \Pi) \text{ s.t. } \forall N \in \mathbb{N} \quad \forall \{q_i\}_{i=1}^N \quad \forall i : y_i = f(q_i). \quad (1)$$

Examples. Consider the problem of computing a maximal independent set. The CENTLOCAL-algorithm is input a sequence of queries $\{q_i\}_i$, each of which is a vertex. For each q_i , the algorithm outputs whether q_i is in I , for an arbitrary yet fixed maximal independent set $I \subseteq V$. Consistency means that I is fixed for all queries. The algorithm has to satisfy this specification even though it does not probe all of G , and obviously does not store the maximal independent set I . Moreover, a stateless algorithm does not even remember the answers it gave to previous queries. Note that if a vertex is queried twice, then the algorithm must return the same answer. If two queries are neighbors, then the algorithm may not answer that both are in the independent set. If the algorithm answers that q_i is not in the independent set, then there must exist a neighbor of q_i for which the algorithm would answer affirmatively. If all vertices are queried, then the answers constitute the maximal independent set I .

As another example, consider the problem of computing a $(\Delta + 1)$ vertex coloring. Consistency in this example means the following. The online algorithm is input a sequence of queries, each of which is a vertex. The algorithm must output the color of each queried vertex. If a vertex is queried twice, then the algorithm must return the same color. Moreover, queried vertices that are neighbors must be colored by different colors. Thus, if all vertices are queried, then the answers constitute a legal vertex coloring that uses $(\Delta + 1)$ colors.

Resources and Performance Measures. The resources used by a CENTLOCAL-algorithm are: probes, space, and random bits. The running time used to answer a query is not counted. The main performance measure is the *maximum number of probes* that the CENTLOCAL-algorithm performs per query. We consider an additional measure called *probe radius*. The probe radius of a CENTLOCAL-algorithm C is r if, for every query q , all the probes that algorithm C performs in G are contained in the ball of radius r centered at q . We denote the probe radius of algorithm A over graph G by $r_G(A)$.

The *state* of algorithm ALG is the information that ALG saves between queries. The *space* of algorithm ALG is the maximum number of bits required to encode the state of ALG. A CENTLOCAL-algorithm is *stateless* if the algorithm does not store any information between queries. In particular, a stateless algorithm does not store previous queries, answers to previous probes, or answers given to previous queries.⁴ In this paper all our CENTLOCAL-algorithms are stateless.

Definition 1. An online algorithm is a CENTLOCAL $[q, s]$ algorithm for Π if (1) it is consistent with (G, Π) , (2) it performs at most q probes, and (3) the space of the algorithm is bounded by s .

The goal in designing algorithms in the CENTLOCAL model is to minimize the number of probes and the space (in particular $q, s = o(n)$). A CENTLOCAL $[q, s]$ algorithm with $s = 0$ is called a stateless CENTLOCAL $[q]$ algorithm. Stateless algorithms are useful in the case of uncoordinated distributed servers that answer queries without communicating with each other.

⁴We remark that in [RTVX11] no distinction was made between the space needed to answer a query and the space needed to store the state between queries. Our approach is different and follows the DISTLOCAL model in which one does not count the space and running time of the vertices during the execution of the distributed algorithm. Hence, we ignore the space and running time of the CENTLOCAL-algorithm during the processing of a query. Interestingly, the state between queries in [ARVX12, MRVX12, MV13, RV14] only stores a random seed that is fixed throughout the execution of the algorithm.

Randomized local algorithms. A randomized CENTLOCAL-algorithm is also parameterized by the *failure probability* δ . We say that ALG is a CENTLOCAL $[q, s, \delta]$ algorithm for Π if the algorithm is consistent, performs at most q probes, and uses space at most s with probability at least $1 - \delta$. The standard requirement is that $\delta = 1/\text{poly}(n)$.

The number of random bits used by a randomized algorithm is also a resource. One can distinguish between two types of random bits: (1) random bits that the algorithm must store between queries, and (2) random bits that are not stored between queries. We use the convention that information that is stored between queries is part of the state. Hence, random bits, even though chosen before the first query, are included in the state if they are stored between queries.⁵

Parallelizability and query order obliviousness. In [ARVX12, MRVX12, MV13] two requirements are introduced: *parallelizability* and *query order obliviousness*. These requirements are fully captured by the definition of a consistent, online, deterministic algorithm with zero space. That is, every online algorithm that is consistent, stateless, and deterministic is both parallelizable and query order oblivious.

2.3 The DISTLOCAL Model

The model of local distributed computation is a classical model (e.g., [Lin92, Pel00, Suo13]). A distributed computation takes place in an undirected labeled graph $G = (V, E)$. The neighbors of each vertex v are numbered from 1 to $\deg(v)$ in an arbitrary but fixed manner. Ports are used to point to the neighbors of v ; the i th port points to the i th neighbor. Each vertex models a processor, and communication is possible only between neighboring processors. Initially, every $v \in V$ is input a local input. The computation is done in $r \in \mathbb{N}$ synchronous rounds as follows. In every round: (1) every processor receives a message from each neighbor, (2) every processor performs a computation based on its local input and the messages received from its neighbors, (3) every processor sends a message to each neighbor. We assume that a message sent in the end of round i is received in the beginning of round $i + 1$. After the r th round, every processor computes a local output.

The following assumptions are made in the DISTLOCAL model: (1) The local input to each vertex v includes the ID of v , the degree of the vertex v , the maximum degree Δ , the number of vertices n , and the ports of v to its neighbors. (2) The IDs are distinct and bounded by a polynomial in n . (3) The length of the messages sent in each round is not bounded. (4) The computation in each vertex in each round need not be efficient.

We say that a distributed algorithm is a DISTLOCAL $[r]$ -algorithm if the number of communication rounds is r . Strictly speaking, a distributed algorithm is considered *local* if r is bounded by a constant. We say that a DISTLOCAL $[r]$ -algorithm is *almost local* if $r = O(\log^*(n))$. When it is obvious from the context we refer to an almost DISTLOCAL algorithm simply by a DISTLOCAL algorithm.

2.4 Mutual Simulations Between DISTLOCAL and CENTLOCAL

In this section we show that one can simulate algorithms over labeled graphs in one model by algorithms in the other model (without any restriction on Δ). Since our algorithms are deterministic, we focus on simulations of deterministic algorithms.

⁵As noted in Footnote 4, in [ARVX12, MRVX12, MV13] the state does not change during the execution of the CENTLOCAL algorithm.

The following definition considers CENTLOCAL-algorithms whose queries are vertices of a graph. The definition can be easily extended to edge queries.

Definition 2. A CENTLOCAL-algorithm C simulates (or is simulated by) a DISTLOCAL-algorithm D if, for every vertex v , the local output of D in vertex v equals the answer that algorithm C computes for the query v .

Simulation of DISTLOCAL by CENTLOCAL [PR07]: Every deterministic DISTLOCAL $[r]$ -algorithm, can be simulated by a deterministic, stateless CENTLOCAL $[O(\Delta^r)]$ -algorithm. The simulation proceeds simply by probing all vertices in the ball of radius r centered at the query. If $\Delta = 2$, then balls are simple paths (or cycles) and hence simulation of a DISTLOCAL $[r]$ -algorithm is possible by a CENTLOCAL $[2r]$ -algorithm.

Simulation of CENTLOCAL by DISTLOCAL: The following Proposition suggests a design methodology for distributed algorithms. For example, suppose that we wish to design a distributed algorithm for maximum matching. We begin by designing a CENTLOCAL-algorithm C for computing a maximum matching. Let r denote the probe radius of the CENTLOCAL-algorithm C . The proposition tells us that we can compute the same matching (that is computed by C) by a distributed r -round algorithm.⁶

Proposition 1. Every stateless deterministic CENTLOCAL-algorithm C whose probe radius is at most r can be simulated by a deterministic DISTLOCAL $[r]$ -algorithm D .

Proof. The distributed algorithm D collects, for every v , all the information in the ball of radius r centered at v . (This information includes the IDs of the vertices in the ball and the edges between them.)

After this information is collected, the vertex v locally runs the CENTLOCAL-algorithm C with the query v . Because algorithm C is stateless, the vertex has all the information required to answer every probe of C . \square

3 Acyclic Orientation

In this section we deal with orientation of undirected graphs, namely, assigning directions to the edges. We suggest to obtain an orientation from a vertex coloring.

Definitions. An *orientation* of an undirected graph $G = (V, E)$ is a directed graph $H = (V, A)$, where $\{u, v\} \in E$ if and only if $(u, v) \in A$ or $(v, u) \in A$ but not both. An orientation H is *acyclic* if there are no directed closed paths in H . The *radius* of an acyclic orientation H is the length of the longest directed path in H . We denote the radius of an orientation by $rad(H)$. In the problem of acyclic orientation with bounded radius (O-RAD), the input is an undirected graph. The output is an orientation H of G that is acyclic. The goal is to compute an acyclic orientation H of G that minimizes $rad(H)$.

The set of vertices that are reachable from v in a directed graph H is called the *reachability set* of v . We denote the reachability set of $v \in V$ in digraph H by $Reach_H(v)$. Let $reach_H(v) \triangleq |Reach_H(v)|$ and $reach(H) \triangleq \max_{v \in V} reach_H(v)$. We simply write $Reach(v)$, $reach(v)$ when

⁶Message lengths grow at a rate of $O(\Delta^{r+1} \cdot \log n)$ as information (e.g., IDs and existence of edges) is accumulated.

the digraph H is obvious from the context. In the problem of acyclic orientation with bounded reachability (OBR), the input is an undirected graph. The output is an orientation H of G that is acyclic. The goal is to minimize $\text{reach}(H)$.

Previous works obtain an acyclic orientation by random vertex ranking [NO08, YY12, ARVX12, MRVX12, MV13]. We propose to obtain an acyclic orientation by vertex coloring.

Proposition 2 (Orientation via coloring). *Every coloring by c colors induces an acyclic orientation with*

$$\text{rad}(H) \leq c - 1.$$

Hence, every CENTLOCAL $[q]$ -algorithm for vertex coloring also implies a CENTLOCAL $[2q]$ -algorithm for acyclic orientation.

Proof. Direct each edge from a high color to a low color. By monotonicity the orientation is acyclic. Every directed path has at most c vertices, and hence the reachability is bounded as required. To determine the orientation of an edge (u, v) , the CENTLOCAL-algorithm simply computes the colors of u and v . \square

The following proposition bounds the maximum cardinality of a reachability by a function of the reachability radius.

Proposition 3.

$$\text{reach}(H) \leq 1 + \Delta \cdot \sum_{i=1}^{\text{rad}(H)} (\Delta - 1)^{i-1} \leq \begin{cases} 2\Delta \cdot (\Delta - 1)^{\text{rad}(H)-1}, & \text{if } \Delta \geq 3, \\ 2 \cdot \text{rad}(H) + 1, & \text{if } \Delta = 2. \end{cases}$$

3.1 A CENTLOCAL Algorithm for Vertex Coloring

In this section we present a deterministic, stateless CENTLOCAL $[O(\Delta \cdot \log^* n + \Delta^3)]$ -algorithm that computes a vertex coloring that uses $c = O(\Delta^2)$ colors (see Theorem 7). Orientation by this coloring yields an acyclic orientation H with $\text{rad}(H) \leq \Delta^2$ and $\text{reach}(H) \leq 2 \cdot \Delta^c$.

CENTLOCAL-algorithms for vertex coloring can be also obtained by simulating DISTLOCAL vertex coloring algorithms. Consider, for example, the $(\Delta + 1)$ coloring using $r_1 = O(\Delta) + \frac{1}{2} \cdot \log^* n$ rounds of [BE09] or the $O(\Delta^2)$ coloring using $r_2 = O(\log^* n)$ rounds of [Lin92]. CENTLOCAL simulations of these algorithms require $O(\Delta^{r_i})$ probes. Thus, in our algorithm, the number of probes grows (slightly) slower as a function of n and is polynomial in Δ .

Our algorithm relies on techniques from two previous DISTLOCAL coloring algorithms.

Theorem 4 ([Lin92, Corollary 4.1]). *A $5\Delta^2 \log c$ coloring can be computed from a c coloring by a DISTLOCAL $[1]$ -algorithm.*

Lemma 5 (Linial 92, Lemma 4.2). *A $O(\Delta^2)$ -coloring can be computed from a $O(\Delta^3)$ -coloring by a DISTLOCAL $[1]$ -algorithm.*

Theorem 6 ([PR01, Section 4]). *A $(\Delta + 1)$ coloring can be computed by a DISTLOCAL $[O(\Delta^2 + \log^* n)]$ -algorithm.*

Theorem 7. *An $O(\Delta^2)$ coloring can be computed by a deterministic, stateless CENTLOCAL $[O(\Delta \cdot \log^* n + \Delta^3)]$ -algorithm. The probe radius of this algorithm is $O(\log^* n)$.*

Proof. We begin by describing a two phased $\text{DISTLOCAL}[O(\log^* n)]$ -algorithm D that uses $O(\Delta^2)$ colors. Algorithm D is especially designed so that it admits an “efficient” simulation by a CENTLOCAL -algorithm.

Consider a graph $G = (V, E)$ with a maximum degree Δ . In the first phase, the edges are partitioned into Δ^2 parts, so that the maximum degree in each part is at most 2. Let $p_i(u)$ denote the neighbor of vertex u pointed to by the i th port of u . Following Kuhn [Kuh09] we partition the edge set E as follows. Let $E_{\{i,j\}} \subseteq E$ be defined by

$$E_{\{i,j\}} \triangleq \{\{u, v\} \mid p_i(u) = v, p_j(v) = u\}.$$

Each edge belongs to exactly one part $E_{\{i,j\}}$. For each part $E_{\{i,j\}}$ and vertex u , at most two edges in $E_{\{i,j\}}$ are incident to u . Hence, the maximum degree in each part is at most 2. Each vertex can determine in a single round how the edges incident to it are partitioned among the parts. Let $G_{\{i,j\}}$ denote the undirected graph over V with edge set $E_{\{i,j\}}$.

By Theorem 6, we 3-color each graph $G_{\{i,j\}}$ in $O(\log^* n)$ rounds. This induces a vector of Δ^2 colors per vertex, hence a 3^{Δ^2} vertex coloring of G .

In the second phase, Algorithm D applies Theorem 4 twice, followed by an application of Theorem 5, to reduce the number of colors to $O(\Delta^2)$.

We now present an efficient simulation of algorithm D by a CENTLOCAL -algorithm C . Given a query for the color of vertex v , Algorithm C simulates the first phase of D in which a 3-coloring algorithm is executed in each part $E_{\{i,j\}}$. Since the maximum degree of each $G_{\{i,j\}}$ is two, a ball of radius r in $G_{\{i,j\}}$ contains at most $2r$ edges. In fact, this ball can be recovered by at most $2r$ probes. It follows that a CENTLOCAL simulation of the 3-coloring of $G_{\{i,j\}}$ performs only $O(\log^* n)$ probes. Observe that if vertex v is isolated in $G_{\{i,j\}}$, then it may be colored arbitrarily (say, by the first color). A vertex v is not isolated in at most Δ parts. It follows that the simulation of the first phase performs $O(\Delta \cdot \log^* n)$ probes.

The second phase of algorithm D requires an additional Δ^3 probes, and the theorem follows. \square

The following corollary is a direct consequence of the coloring algorithm described in Theorem 7, the orientation induced by the coloring in Proposition 2, and the bound on the reachability based on the radius in Proposition 3.

Corollary 8. *There is a deterministic, stateless $\text{CENTLOCAL}[O(\Delta \cdot \log^* n + \Delta^3)]$ -algorithm for orienting a graph that achieves $\text{rad}(H) \leq \Delta^2$ and $\text{reach}(H) \leq \Delta^{O(\Delta^2)}$.*

4 Deterministic Localization of Sequential Algorithms and Applications

A common theme in online algorithms and “greedy” algorithms is that the elements are scanned in query order or in an arbitrary order, and a decision is made for each element based on the decisions of the previous elements. Classical examples of such algorithms include the greedy algorithms for maximal matchings, $(\Delta + 1)$ vertex coloring, and maximal independent set. We present a compact and axiomatic CENTLOCAL deterministic simulation of this family of algorithms, for which a randomized simulation appeared in [MRVX12]. Our deterministic simulation is based on an acyclic orientation that induces a partial order.

For simplicity, consider a graph problem Π , the solution of which is a function $g(v)$ defined over the vertices of the input graph. For example, $g(v)$ can be the color of v or a bit indicating if v belongs to a maximal independent set. (One can easily extend the definition to problems in which the solution is a function over the edges, e.g., maximal matching.)

We refer to an algorithm as a *sequential algorithm* if it fits the scheme listed as Algorithm 1. The algorithm $\text{ALG}(G, \sigma)$ is input a graph $G = (V, E)$ and a bijection $\sigma : \{1, \dots, n\} \rightarrow V$ of the vertices. The bijection σ orders the vertices in total order, if $\sigma(i) = v$ then v is the i th vertex in the order and $\sigma^{-1}(v) = i$. The algorithm scans the vertices in the order induced by σ . It determines the value of $g(\sigma(i))$ based on the values of its neighbors whose value has already been determined. This decision is captured by the function f in Line 2. For example, in vertex coloring, f returns the smallest color that does not appear in a given a subset of colors.

Algorithm 1 The sequential algorithm scheme.

Input: A graph $G = (V, E)$ and a bijection $\sigma : \{1, \dots, n\} \rightarrow V$.

- 1: **for** $i = 1$ to n **do**
 - 2: $g(\sigma(i)) \leftarrow f(\{g(v) : v \in \Gamma(\sigma(i)) \ \& \ \sigma^{-1}(v) < i\})$ \triangleright (Decide based on “previous” neighbors)
 - 3: **end for**
 - 4: **Output:** g .
-

Lemma 9. *Let $G = (V, E)$ be a graph, let $H = (V, A)$ be an acyclic orientation of G and let $P_{>} \subseteq V \times V$ denote the partial order defined by the transitive closure of H . Namely, $(u, v) \in P_{>}$ if and only if there exists a directed path from u to v in H . Let ALG denote a sequential algorithm. For every bijection $\sigma : \{1, \dots, n\} \rightarrow V$ that is a linear extension of $P_{>}$ (i.e., for every $(u, v) \in P_{>}$ we have that $\sigma^{-1}(u) > \sigma^{-1}(v)$), the output of $\text{ALG}(G, \sigma)$ is the same.*

Proof. Consider two linear extensions σ and τ of $P_{>}$. Let g_σ denote the output of $\text{ALG}(G, \sigma)$ and define g_τ analogously.

Let

$$B_\sigma(u) \triangleq \{v \in \Gamma(u) \mid \sigma^{-1}(u) > \sigma^{-1}(v)\}.$$

We claim that $B_\sigma(u) = B_\tau(u)$ for every u . By symmetry, it suffices to prove that $B_\sigma(u) \subseteq B_\tau(u)$. Consider a vertex $v \in B_\sigma(u)$. We need to show that $v \in B_\tau(u)$. By definition, v is a neighbor of u . We consider the two possible orientations of the edge (u, v) . If $(u, v) \in A$, then $(u, v) \in P_{>}$. Hence $\sigma^{-1}(u) > \sigma^{-1}(v)$ and $\tau^{-1}(u) > \tau^{-1}(v)$ because σ and τ are linear extensions of $P_{>}$. We conclude that $v \in B_\tau(u)$, as required. If $(v, u) \in A$, then $\sigma^{-1}(u) < \sigma^{-1}(v)$, and this implies that $v \notin B_\sigma(u)$, a contradiction.

To complete the proof, we prove by induction on i that for $u = \sigma^{-1}(i)$ we have $g_\sigma(u) = g_\tau(u)$. Indeed, $g_\sigma(u) = f(B_\sigma(u))$ and $g_\tau(u) = f(B_\tau(u))$. For $i = 1$ we have $B_\sigma(u) = B_\tau(u) = \emptyset$, hence $f(B_\sigma(u)) = f(B_\tau(u))$, as required. To induction step recall that $B_\sigma(u) = B_\tau(u)$. By the induction hypothesis we conclude that $f(B_\sigma(u)) = f(B_\tau(u))$, and the lemma follows. \square

The following theorem proves that a sequential algorithm can be simulated by a $\text{CENTLOCAL}[q]$ -algorithm. The number of probes q equals the number of probes used by the vertex coloring algorithm (that induces an acyclic orientation) times the max-reachability of the orientation.

Theorem 10. *For every sequential algorithm ALG , there exists a deterministic, stateless $\text{CENTLOCAL}[\Delta^{O(\Delta^2)} \cdot \log^* n]$ -algorithm ALG_c that simulates ALG in the following sense. For every graph G , there exists a bijection σ , such that $\text{ALG}_c(G)$ simulates $\text{ALG}(G, \sigma)$. That is, for every vertex v in G , the answer of $\text{ALG}_c(G)$ on query v is $g_\sigma(v)$, where g_σ denotes the output of $\text{ALG}(G, \sigma)$.*

Proof. Consider the acyclic orientation H of G computed by the $\text{CENTLOCAL}[\Delta \cdot \log^* n + \Delta^3]$ -algorithm presented in Corollary 8. Let P_{\succ} denote the partial order that is induced by H , and let σ be any linear extension of P_{\succ} (as defined in Lemma 9). On query $v \in V$ the value $g_\sigma(v)$ is computed by performing a (directed) DFS on H that traverses the subgraph of H induced by $R_H(v)$. The DFS uses the CENTLOCAL algorithm from Corollary 8 to determine the orientation of each incident edge and continues only along outward-directed edges⁷. The value of $g_\sigma(v)$ is determined when the DFS backtracks from v . The product of $\text{reach}(H) = \Delta^{O(\Delta^2)}$ and the number of probes of the orientation algorithm bounds the number of probes of ALG_c . Hence, we obtain that $\Delta^{O(\Delta^2)} \cdot \log^* n$ probes suffice, and the theorem follows. \square

Corollary 11. *There are deterministic, stateless $\text{CENTLOCAL}[\Delta^{O(\Delta^2)} \cdot \log^* n]$ algorithms for $(\Delta + 1)$ -vertex coloring, maximal independent set, and maximal matching.*

We have described two CENTLOCAL coloring algorithms; one uses Δ^2 colors (Theorem 7), and the second uses $\Delta + 1$ colors (Corollary 11). The number of probes of the $(\Delta + 1)$ -coloring obtained by simulating the sequential coloring algorithm is exponential in Δ . The Δ^2 -coloring algorithm requires only $O(\Delta \cdot \log^* n + \Delta^3)$ probes. Hence, increasing the number of colors (by a factor of Δ) enables us to reduce the dependency of the number of probes on the maximum degree.

We conclude with the following immediate lemma that bounds the probe radius of the CENTLOCAL -algorithm for MIS.

Lemma 12. *Let AO denote a stateless CENTLOCAL -algorithm that computes an acyclic orientation $H = (V, A)$ of a graph $G = (V, E)$. Let r denote the probe radius of AO . Then, there exists a stateless CENTLOCAL -algorithm for MIS whose probe radius is at most $r + \text{rad}(H)$.*

Assume that the acyclic orientation is based on the $\text{CENTLOCAL}[O(\Delta \cdot \log^* n + \Delta^3)]$ -algorithm that computes a Δ^2 -vertex coloring. The probe radius of the MIS-algorithm implied by lemma 12 is $O(\log^* n + \Delta^2)$. Indeed, the probes of the Δ^2 -coloring algorithm are confined to a ball of radius $O(\log^* n)$. The probes of the simulation of the sequential algorithm are confined to a ball of radius $c = O(\Delta^2)$.

Let L-MIS denote the CENTLOCAL algorithm for maximal independent set (MIS) stated in Corollary 11. The Boolean predicate $\text{L-MIS}(G, v)$ indicates if v is in the MIS of G computed by Algorithm L-MIS.

5 A CENTLOCAL Approximate MCM Algorithm

In this section we present a stateless deterministic CENTLOCAL algorithm that computes a $(1 - \varepsilon)$ -approximation of a maximum cardinality matching. The algorithm is based on a CENTLOCAL -algorithm for maximal independent set (see Corollary 11) and on the local improvement technique of Nguyen and Onak [NO08].

⁷Given that the CENTLOCAL algorithm works by running a CENTLOCAL coloring algorithm, one can actually use the latter algorithm directly.

Terminology and Notation. Let M be a matching in $G = (V, E)$. A vertex $v \in V$ is M -free if v is not an endpoint of an edge in M . A simple path is M -alternating if it consists of edges drawn alternately from M and from $E \setminus M$. A path is M -augmenting if it is M -alternating and if both of the path's endpoints are M -free vertices. Note that the length of an augmenting path must be odd. The set of edges in a path p is denoted by $E(p)$, and the set of edges in a collection P of paths is denoted by $E(P)$. Let $A \oplus B$ denote the symmetric difference of the sets A and B .

Description of The Global Algorithm. Similarly to [LPSP08, NO08, MV13] our local algorithm simulates the global algorithm listed as Algorithm 2. This global algorithm builds on lemmas of Hopcroft and Karp [HK73] and Nguyen and Onak [NO08].

Lemma 13 ([HK73]). *Let M be a matching in a graph G . Let k denote the length of a shortest M -augmenting path. Let P^* be a maximal set of vertex disjoint M -augmenting paths of length k . Then, $(M \oplus E(P^*))$ is a matching and the length of every $(M \oplus E(P^*))$ -augmenting path is at least $k + 2$.*

Lemma 14 ([NO08, Lemma 6]). *Let M^* be a maximum matching and M be a matching in a graph G . Let $2k + 1$ denote the length of a shortest M -augmenting path. Then*

$$|M| \geq \frac{k}{k+1} \cdot |M^*|.$$

Algorithm 2 Global-APX-MCM(G, ε).

Input: A graph $G = (V, E)$ and $0 < \varepsilon < 1$.
Output: A $(1 - \varepsilon)$ -approximate matching

- 1: $M_0 \leftarrow \emptyset$.
- 2: $k \leftarrow \lceil \frac{1}{\varepsilon} \rceil$.
- 3: **for** $i = 0$ to k **do**
- 4: $P_{i+1} \leftarrow \{p \mid p \text{ is an } M_i\text{-augmenting path, } |p| = 2i + 1\}$.
- 5: $P_{i+1}^* \subseteq P_{i+1}$ is a maximal vertex disjoint subset of paths.
- 6: $M_{i+1} \triangleq M_i \oplus E(P_{i+1}^*)$.
- 7: **end for**
- 8: **Return** M_{k+1} .

Algorithm 3 Global-APX-MCM'(G, ε).

Input: A graph $G = (V, E)$ and $0 < \varepsilon < 1$.
Output: A $(1 - \varepsilon)$ -approximate matching

- 1: $M_0 \leftarrow \emptyset$.
- 2: $k \leftarrow \lceil \frac{1}{\varepsilon} \rceil$.
- 3: **for** $i = 0$ to k **do**
- 4: Construct the intersection graph H_i over P_i .
- 5: $P_{i+1}^* \leftarrow \text{MIS}(H_i)$.
- 6: $M_{i+1} \triangleq M_i \oplus E(P_{i+1}^*)$.
- 7: **end for**
- 8: **Return** M_{k+1} .

Algorithm 2 is given as input a graph G and an approximation parameter $\varepsilon \in (0, 1)$. The algorithm works in $k + 1$ iterations, where $k = \lceil \frac{1}{\varepsilon} \rceil$. Initially, $M_0 = \emptyset$. The invariant of the algorithm is that M_i is a matching, every augmenting path of which has length at least $2i + 1$. Given M_i , a new matching M_{i+1} is computed as follows. Let P_{i+1} denote the set of shortest M_i -augmenting paths. Let $P_{i+1}^* \subseteq P_{i+1}$ denote a maximal subset of vertex disjoint paths. Define $M_{i+1} \triangleq M_i \oplus E(P_{i+1}^*)$. By Lemmas 13 and 14, we obtain the following result.

Theorem 15. *The matching M_{k+1} computed by Algorithm 2 is a $(1 - \varepsilon)$ -approximation of a maximum matching.*

The intersection graph. Define the intersection graph $H_i = (P_i, C_i)$ as follows. The set of nodes P_i is the set of M_{i-1} -augmenting paths of length $2i - 1$. We connect two paths p and q in P_i by an edge $(p, q) \in C_i$ if p and q intersect (i.e., share a vertex in V). Note that H_1 is the line graph of G and that M_1 is simply a maximal matching in G . Observe that P_i^* as defined above is a maximal independent set in H_i . Thus, iteration i of the global algorithm can be conceptualized by the following steps (see Algorithm 3): construct the intersection graph H_i , compute a maximal independent set P_i^* in H_i , and augment the matching by $M_i \triangleq M_{i-1} \oplus (E(P_i^*))$.

Implementation by a stateless deterministic CENTLOCAL Algorithm. The recursive local improvement technique in [NO08, Section 3.3] simulates the global algorithm. It is based on a recursive oracle \mathcal{O}_i . The input to oracle \mathcal{O}_i is an edge $e \in E$, and the output is a bit that indicates whether $e \in M_i$. Oracle \mathcal{O}_i proceeds by computing two bits τ and ρ (see Algorithm 4). The bit τ indicates whether $e \in M_{i-1}$, and is computed by invoking oracle \mathcal{O}_{i-1} . The bit ρ indicates whether $e \in E(P_i^*)$ (where P_i^* is an MIS in H_{i-1}). Oracle \mathcal{O}_i returns $\tau \oplus \rho$ because $M_i = M_{i-1} \oplus E(P_i^*)$.

We determine whether $e \in E(P_i^*)$ by running the CENTLOCAL-algorithm \mathcal{A}_i over H_i (see Algorithm 5). Note that \mathcal{A}_1 simply computes a maximal matching (i.e., a maximal independent set of the line graph H_1 of G). The main difficulty we need to address is how to simulate the construction of H_i and probes to vertices in H_i . We answer the question whether $e \in E(P_i^*)$ by executing the following steps: (1) Listing: construct the set $P_i(e) \triangleq \{p \in P_i \mid e \in E(p)\}$. Note that $e \in E(P_i^*)$ if and only if $P_i(e) \cap P_i^* \neq \emptyset$. (2) MIS-step: for each $p \in P_i(e)$, input the query p to an MIS-algorithm for H_i to test whether $p \in P_i^*$. If an affirmative answer is given to one of these queries, then we conclude that $e \in E(P_i^*)$. We now elaborate on how the listing step and the MIS-step are carried out by a CENTLOCAL-algorithm.

The listing of all the paths in $P_i(e)$ uses two preprocessing steps: (1) Find the balls of radius $2i - 1$ in G centered at the endpoints of e . (2) Check if $e' \in M_{i-1}$ for each edge e' incident to vertices in the balls. We can then exhaustively check for each path p of length $2i - 1$ that contains e whether p is an M_{i-1} -augmenting path.

The MIS-step answers a query $p \in P_i^*$ by simulating the MIS CENTLOCAL-algorithm over H_i . The MIS-algorithm needs to simulate probes to H_i . A probe to H_i consists of an M_{i-1} -augmenting path q and a port number. We suggest to implement this probe by probing all the neighbors of q in H_i (so the port number does not influence the first part of implementing a probe). See Algorithm 6. As in the listing step, a probe q in H_i can be obtained by (1) finding the balls in G of radius $2i - 1$ centered at endpoints of edges in $E(q)$, and (2) finding out which edges within these balls are in M_{i-1} . The first two steps enable us to list all of the neighbors of q in H_i (i.e., the M_{i-1} -augmenting paths that intersect q). These neighbors are ordered (e.g.,

by lexicographic order of the node IDs along the path). If the probe asks for the neighbor of q in port i , then the implementation of the probe returns the i th neighbor of q in the ordering.

By combining the recursive local improvement technique with our deterministic stateless CENTLOCAL MIS-algorithm, we obtain a deterministic stateless CENTLOCAL-algorithm that computes a $(1 - \varepsilon)$ -approximation for maximum matching. The algorithm is invoked by calling the oracle \mathcal{O}_{k+1} .

Lemma 16. *The oracle $\mathcal{O}_i(e)$ is a CENTLOCAL $[2^{\Delta^{O(i)}} \cdot (\log^* n)^i]$ that computes whether $e \in M_i$.*

Proof. Correctness follows by induction on i that shows that the oracle simulates Algorithm 3. We analyze the number of probes as follows. To simplify notation, we denote the number of probes performed by algorithm B by $|B|$, for example, $|\mathcal{O}_i|$ and $|\mathcal{A}_i|$ denote the number of probes to G performed by the oracle \mathcal{O}_i and procedure \mathcal{A}_i , respectively. Let n_i and Δ_i denote the number of vertices and the maximum degree of H_i , respectively.

The probe complexity of \mathcal{O}_i satisfies the following recurrence:

$$|\mathcal{O}_i| = \begin{cases} 0 & \text{if } i = 0, \\ |\mathcal{O}_{i-1}| + |\mathcal{A}_i| & \text{if } i \geq 1. \end{cases}$$

The probe complexity of \mathcal{A}_i is upper bounded as follows. In Lines 2-3, each BFS performs $O(\Delta^{2i})$ probes. The number of edges in the probed ball is $O(\Delta^{2i+1})$, and for each such edge a call to \mathcal{O}_{i-1} is made in Line 4. Line 5 does not generate any probes. Let $|\text{L-MIS}_G(H_i)|$ denote the probe complexity of the simulation of CENTLOCAL-algorithm for MIS over the intersection graph H_i when the access is to G . In Line 7, the number of probes is bounded by $|P_i(e)| \cdot |\text{L-MIS}_G(H_i)|$. Hence,

$$|\mathcal{A}_i| \leq O(\Delta^{2i+1}) \cdot |\mathcal{O}_{i-1}| + |P_i(e)| \cdot |\text{L-MIS}_G(H_i)|.$$

The number of paths in $P_i(e)$ is at most $2i \cdot \Delta^{2i}$ (indeed, there are $2i$ possibilities for the position of e along a path, and, for each position j , there are at most $\Delta^j \cdot \Delta^{2i-j}$ paths p such that e is the j th edge in p).

We bound $|\text{L-MIS}_G(H_i)|$ by the probe complexity $|\text{L-MIS}_{H_i}(H_i)|$ (namely, the probe complexity if one can access H_i) times the probe complexity of simulating probes to H_i via probes to G . By Corollary 11, $|\text{L-MIS}_{H_i}(H_i)| \leq \Delta_i^{O(\Delta_i^2)} \cdot \log^* n_i$. Simulation of probes in H_i via probes to G is implemented by the $\text{probe}(i, p)$ procedure. Similarly, to the analysis of the probe complexity of \mathcal{A}_i , the probe complexity of $\text{probe}(i, p)$ is $O(2i \cdot \Delta^{2i+1} \cdot |\mathcal{O}_{i-1}|)$.

Hence,

$$|P_i(e)| \cdot |\text{L-MIS}_G(H_i)| \leq 2i \cdot \Delta^{2i} \cdot \Delta_i^{O(\Delta_i^2)} \cdot \log^* n_i \cdot 2i \cdot \Delta^{2i+1} \cdot |\mathcal{O}_{i-1}|.$$

Because $n_i \leq n^{2i}$ and $\Delta_i = O(i^2 \cdot \Delta^{2i})$, it follows that

$$|\mathcal{A}_i| \leq \Delta^{\Delta^{O(i)}} \cdot \log^* n \cdot |\mathcal{O}_{i-1}|.$$

We conclude that $|\mathcal{O}_i|$ satisfies

$$\begin{aligned} |\mathcal{O}_i| &\leq \Delta^{\Delta^{O(i)}} \cdot \log^* n \cdot |\mathcal{O}_{i-1}| \\ &\leq \Delta^{\Delta^{O(i)}} \cdot (\log^* n)^i. \end{aligned}$$

Note that $\Delta^{\Delta^{O(i)}} = 2^{\Delta^{O(i)}}$, and the lemma follows. \square

By setting $i = \lceil \frac{1}{\varepsilon} \rceil + 1$, we obtain the following theorem.

Theorem 17. *There is a deterministic, stateless, $(1 - \varepsilon)$ -approximate CENTLOCAL $[\varphi]$ -algorithm for maximum matching, where*

$$\varphi = (\log^* n)^{\lceil \frac{1}{\varepsilon} \rceil + 1} \cdot 2^{\Delta^{O(1/\varepsilon)}}.$$

Algorithm 4 $\mathcal{O}_i(e)$ - a recursive oracle for membership in the approximate matching.

Input: A query $e \in E$.

Output: Is e an edge in the matching M_i ?

- 1: If $i = 0$ then return false.
 - 2: $\tau \leftarrow \mathcal{O}_{i-1}(e)$.
 - 3: $\rho \leftarrow \mathcal{A}_i(e)$.
 - 4: **Return** $\tau \oplus \rho$.
-

Algorithm 5 $\mathcal{A}_i(e = (u, v))$ - a procedure for checking membership of an edge e in one of the paths in P_i^* .

Input: An edge $e \in E$.

Output: Does e belong to a path $p \in P_i^*$?

- 1: **Listing step:** \triangleright Compute all shortest M_{i-1} -augmenting paths that contain e .
 - 2: $B_u \leftarrow BFS_G(u)$ with depth $2i - 1$.
 - 3: $B_v \leftarrow BFS_G(v)$ with depth $2i - 1$.
 - 4: For every edge e' in the subgraph of G induced by $B_u \cup B_v$: $\chi_{e'} \leftarrow \mathcal{O}_{i-1}(e')$.
 - 5: $P_i(e) \leftarrow$ all M_{i-1} -augmenting paths of length $2i - 1$ that contain e (based on information gathered in Lines 2-4).
 - 6: **MIS-step:** \triangleright Check if one of the augmenting paths is in P_i^* .
 - 7: For every $p \in P_i(e)$: If L-MIS(H_i, p) **Return** true.
 - 8: **Return** false.
-

Algorithm 6 $probe(i, p)$ - simulation of a probe to the intersection graph H_i via probes to G . The probe returns all the M_{i-1} -augmenting paths that intersect p .

Input: A path $p \in P_i$ and the ability to probe G .

Output: The set of M_{i-1} -augmenting paths of length $2i - 1$ that intersect p .

- 1: For every $v \in p$ do
 - 2: $B_v \leftarrow BFS_G(v)$ with depth $2i - 1$.
 - 3: For every edge $e' \in B_v$: $\chi_{e'} \leftarrow \mathcal{O}_{i-1}(e')$. \triangleright determine whether the path is alternating and whether the endpoints are M_{i-1} -free.
 - 4: $P_i(v) \leftarrow$ all M_{i-1} -augmenting paths of length $2i - 1$ that contain v .
 - 5: **Return** $\bigcup_{v \in p} P_i(v)$.
-

6 A DISTLOCAL Approximate MCM Algorithm

In this section, we present a DISTLOCAL-algorithm that computes a $(1 - \varepsilon)$ -approximate maximum cardinality matching. The algorithm is based on bounding the probe radius of the CENTLOCAL-algorithm from Theorem 17 and applying the simulation from Proposition 1.

Theorem 18. *There is a deterministic $\text{DISTLOCAL}[\Delta^{O(1/\varepsilon)} + O(\frac{1}{\varepsilon^2}) \cdot \log^*(n)]$ -algorithm for computing a $(1 - \varepsilon)$ -approximate MCM.*

Proof. The proof of the theorem is based on the simulation of a CENTLOCAL -algorithm by a DISTLOCAL -algorithm from Proposition 1. In Lemma 19 we prove that the probe radius of \mathcal{O}_k is $\Delta^{O(k)} + O(k^2) \cdot \log^*(n)$. Plug $k = 1 + \lceil \frac{1}{\varepsilon} \rceil$, and the theorem follows. \square

Lemma 19. *The probe radius of the CENTLOCAL -algorithm \mathcal{O}_k is*

$$r_G(\mathcal{O}_k) = \Delta^{O(k)} + O(k^2) \cdot \log^*(n).$$

Proof. The probe radius $r_G(\mathcal{O}_i)$ satisfies the following recurrence:

$$r_G(\mathcal{O}_i) = \begin{cases} 0 & \text{if } i = 0, \\ \max\{r_G(\mathcal{O}_{i-1}), r_G(\mathcal{A}_i)\} & \text{if } i \geq 1. \end{cases}$$

The description of the procedure \mathcal{A}_i implies that the probe radius $r_G(\mathcal{A}_i)$ satisfies the following recurrence:

$$r_G(\mathcal{A}_i) \leq \max\{2i + r_G(\mathcal{O}_{i-1}), 2i - 1 + r_G(\text{L-MIS}(H_i))\}$$

We bound the probe radius $r_G(\text{L-MIS}(H_i))$ by composing the radius $r_{H_i}(\text{L-MIS}(H_i))$ with the increase in radius incurred by the simulation of probes to H_i by probes to G . Recall that the L-MIS -algorithm is based on a deterministic coloring algorithm C . We denote the number of colors used by C to color a graph G' by $|C(G')|$.

The MIS -algorithm orients the edges by coloring the vertices. The radius of the orientation is at most the number of colors. It follows that

$$r_{H_i}(\text{L-MIS}(H_i)) \leq r_{H_i}(C(H_i)) + |C(H_i)|.$$

The simulation of probes to H_i requires an increase in the probe radius. In general, suppose that algorithm L probes H , and algorithm S simulates probes to H by probes to G . Let $S(p)$ denote the set of probes in G performed by S to simulate a probe of p in H . Suppose that $S(p) \cap S(p') \neq \emptyset$ whenever p and p' are neighbors in H . In this case the probe radius of the composed algorithm is at most $r_H(L) \cdot r_G(S)$. However, our case is special in the following sense. Consider a path p_1, p_2, \dots, p_r of length r in H_i . This sequence $\{p_j\}$ of probes in H is simulated by probes in G by the procedure $\text{probe}(i, p_j)$, for $j = 1, \dots, r$. The probe radius in G from any vertex in p_1 is bounded by $(2i) \cdot r + r_G(\text{probe}(i - 1))$. Hence,

$$r_G(\text{L-MIS}(H_i)) \leq 2i \cdot r_{H_i}(\text{L-MIS}(H_i)) + r_G(\text{probe}(i - 1)).$$

Many distributed coloring algorithms find a vertex coloring in $O(\log^*(n) + \text{poly}(\Delta))$ rounds (giving us the same upper bound on the probe-radius of the corresponding CENTLOCAL -algorithm) and use $\text{poly}(\Delta)$ colors (see, for example, [BE09, Lin92, CV86, PR01, Kuh09]). Plugging these parameters in the recurrences yields

$$\begin{aligned} r_G(\mathcal{O}_i) &\leq 2i + r_G(\text{L-MIS}(H_i)) \\ &\leq 2i \cdot (1 + r_{H_i}(\text{L-MIS}(H_i))) + r_G(\text{probe}(i - 1)) \\ &\leq r_G(\mathcal{O}_{i-1}) + O\left(i \cdot \log^*(n_i) + \text{poly}(\Delta_i)\right), \end{aligned}$$

Since $\Delta_i \leq (2i)^2 \Delta^{2i-1}$ and $n_i \leq n^{2i}$ we get that

$$\begin{aligned} r_G(\mathcal{O}_k) &\leq \sum_{i=1}^k O(i \cdot \log^*(n) + \text{poly}((2i)^2 \cdot \Delta^{2i})) \\ &= O(k^2 \cdot \log^*(n)) + \Delta^{O(k)}. \end{aligned}$$

The lemma follows. \square

7 A Global $(1 - \varepsilon)$ -Approximate MWM Algorithm

In this section we present a deterministic stateless CENTLOCAL-algorithm that computes a $(1 - \varepsilon)$ -approximation of a maximum weighted matching.⁸ The algorithm is based on a parallel $(1 - \varepsilon)$ -approximation algorithm for weighted matching of Hougardy and Vinkemeier [HV06].

Terminology and Notation. In addition to the terminology and notation used in the unweighted case, we define the following terms. In the weighted case, a path is M -alternating if it is a simple path or a simple cycle in which the edges alternate between M and $E \setminus M$. For a matching M and an M -alternating path p , the *gain* of p is defined by

$$\text{gain}_M(p) \triangleq w(p \setminus M) - w(p \cap M).$$

The gain of a set of (disjoint) paths is the sum of the gains of the paths in the set.

An M -alternating path p is M -augmenting if $\text{gain}_M(p) > 0$ and p satisfies one of the following conditions: (1) p is a simple cycle, or (2) p is a simple path that satisfies: if p ends (or begins) in an edge not in M , then the corresponding endpoint is M -free. Note that the symmetric difference between M and any set of vertex disjoint M -augmenting paths is a matching with higher weight.

We say that a path p is $(M, [1, k])$ -augmenting if p is M -augmenting and $|E(p) \setminus M| \leq k$. An $(M, [1, k])$ -augmenting path may contain at most $2k + 1$ edges (k non-matching edges and $k + 1$ matching edges). The *gain-index* of an M -augmenting path p is defined by

$$\gamma_M(p) \triangleq \lceil \log_2 \text{gain}_M(p) \rceil.$$

Let $I(M)$ denote the intersection graph of $(M, [1, k])$ -augmenting paths. Namely, the vertices of $I(M)$ are the $(M, [1, k])$ -augmenting paths, and two vertices in $I(M)$ are neighbors if they have a common vertex in G . We partition the vertices of $I(M)$ (i.e., $(M, [1, k])$ -augmenting paths of G) to classes; the *class* of an augmenting path equals its gain-index.

Optimal Set of Augmentation Paths. Given a matching M , let $\text{AUG}(M, k)$ denote a set of vertex disjoint $(M, [1, k])$ -augmentation paths with maximum gain. Equivalently, $\text{AUG}(M, k)$ is an MIS in $I(M)$ with maximum gain.

Theorem 20 ([PS04]). *Let M and M^* denote a matching and maximum weight matching in G , respectively, then*

$$\text{gain}(\text{AUG}(M, k)) \geq \frac{k+1}{2k+1} \cdot \left(\frac{k}{k+1} \cdot w(M^*) - w(M) \right).$$

⁸To avoid dealing with constants, we present a $1 - O(\varepsilon)$ -approximation.

Index-Greedy Augmentation. An *index-greedy* set of augmentation paths is an MIS in $I(M)$ obtained by the sequential MIS algorithm where the vertices in $I(M)$ are sorted in non-increasing gain-index order. We denote an index-greedy augmentation by $\text{AUG}^{ig}(M, k)$.

The following proposition states that the gain of every index-greedy augmentation is a $2(k+1)$ -approximation of the gain of $\text{AUG}(M, k)$. It follows from the fact that a greedy MIS is a $(k+1)$ -approximation of a max-weight MIS if each vertex in the greedy MIS intersects at most $(k+1)$ vertices from a max-weight MIS, and from the fact that the ratio between gains of paths with the same gain-index is at most 2.

Proposition 21.

$$\text{gain}(\text{AUG}^{ig}(M, k)) \geq \frac{1}{2(k+1)} \cdot \text{gain}(\text{AUG}(M, k)).$$

Proof. Let $\text{AUG}^{ig}(M, k) = \{p_1, \dots, p_r\}$, where $\gamma_M(p_i) \geq \gamma_M(p_{i+1})$. Namely, p_i is added to the index-greedy augmentation before p_{i+1} . We partition $\text{AUG}(M, k)$ into disjoint sets $X_1 \cup \dots \cup X_r$ as follows. Each augmentation path $q \in \text{AUG}(M, k)$ is in the set X_i with the smallest index i such that $q = p_i$ or q is a neighbor of p_i (in the intersection graph $I(M)$).

Since $X_1 \cup \dots \cup X_r$ is a partition of $\text{AUG}(M, k)$, it suffices to prove that $\min_i \frac{\text{gain}(p_i)}{\text{gain}(X_i)} \geq \frac{1}{2(k+1)}$. Indeed, this inequality follows from two facts. First, every $(M, [1, k])$ -augmenting path intersects at most $k+1$ paths in $\text{AUG}(M, k)$. Second, by the ordering of the augmentations in non-increasing gain-index order, it follows that $\text{gain}(p_i) \geq \frac{1}{2} \cdot \text{gain}(q)$, for every $q \in X(p_i)$. \square

Outline of the Global Algorithm. The main differences between the global approximation algorithms for weighted and unweighted matchings are: (1) The length of the augmenting paths (and cycles) does not grow; instead, during every step, $(M, [1, k])$ -augmenting paths are used. (2) The set of disjoint augmenting paths in each iteration in the weighted case is chosen greedily, giving precedence to augmentations with higher gain-index. We denote the computation of an index-greedy augmentation by IG-MIS. The global algorithm is listed as Algorithm 7.

Algorithm Notation. The global algorithm uses the following notation. The algorithm computes a sequence of matchings M_i (where $i \in [1, L]$, for $L = O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$). We denote the initial empty matching by M_0 . Let $I(M_i)$ denote the intersection graph over $(M_i, [1, k])$ -augmenting paths with edges between paths whenever the paths share a vertex. The class of each vertex in $I(M_i)$ (i.e., augmenting path in G) is the gain-index of the path. Let $\text{IG-MIS}(I(M_i))$ denote a index-greedy MIS in $I(M_i)$ with precedence given to vertices with higher gain-indexes.

Algorithm 10 *probe*($i - 1, p$) - simulation of a probe to the intersection graph $I(M_{i-1})$ via probes to G .

Input: An $(M_{i-1}, [1, k])$ -augmenting path $p \in I(M_{i-1})$ and the ability to probe G .

Output: The set of $(M_{i-1}, [1, k])$ -augmenting paths that intersect p (i.e., neighbors of p in $I(M_{i-1})$).

- 1: For every $v \in p$ do
 - 2: $B_v \leftarrow BFS_G(v)$ with depth $2k + 1$.
 - 3: For every edge $e' \in B_v$: $\chi_e \leftarrow \mathcal{O}_{i-1}(e)$. ▷ needed to determine whether a path is an $(M_{i-1}, [1, k])$ -augmenting path.
 - 4: $P_i(v) \leftarrow$ all $(M_{i-1}, [1, k])$ -augmenting paths that contain v .
 - 5: **Return** $\bigcup_{v \in p} P_i(v)$.
-

Correctness.

Theorem 22 ([HV06]). *Algorithm 7 computes a $(1 - \varepsilon)$ -approximate maximum weighted matching.*

Proof. By Propositions 21 the augmentations computed in Line 6 of the algorithm satisfy

$$gain(\text{AUG}_i) \geq \frac{1}{2(k+1)} \cdot gain(\text{AUG}(M_{i-1}, k)). \quad (2)$$

By Theorem 20

$$gain(\text{AUG}(M_{i-1}, k)) \geq \frac{k+1}{2k+1} \left(\frac{k}{k+1} \cdot w(M^*) - w(M_{i-1}) \right).$$

Let $\rho_i \triangleq w(M_i)/w(M^*)$. It follows that ρ_i satisfies the recurrence

$$\rho_i \geq \left(1 - \frac{1}{2(2k+1)} \right) \rho_{i-1} + \frac{k}{k+1} \cdot \frac{1}{2(2k+1)}.$$

Hence,

$$\begin{aligned} \rho_L &\geq \frac{k}{k+1} \cdot \frac{1}{2(2k+1)} \cdot \frac{1 - \left(1 - \frac{1}{2(2k+1)} \right)^L}{1 - \left(1 - \frac{1}{2(2k+1)} \right)} \\ &= \frac{k}{k+1} \cdot \left(1 - \left(1 - \frac{1}{2(2k+1)} \right)^L \right). \end{aligned}$$

The theorem follows by setting $k = \Theta\left(\frac{1}{\varepsilon}\right)$ and $L = \Theta\left(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon}\right)$. □

8 A CENTLOCAL $(1 - \varepsilon)$ -Approximate MWM Algorithm

In this section we present a CENTLOCAL-algorithm that implements the global $(1 - \varepsilon)$ -approximation algorithm for MWM.

8.1 Preprocessing

We assume that the maximum edge weight is known (as well as n, ε , and Δ). By normalizing the weights, we obtain that the edge weights are in the interval $(0, 1]$. Note, that at least one edge has weight 1.

We round down the edge weights to the nearest integer multiple of ε/n . Let $w(e)$ denote the original edge weights and let $w'(e)$ denote the rounded down weights. Therefore, $w(e) - \varepsilon/n < w'(e) \leq w(e)$. Note that as a result of rounding down edge weights, the minimum positive weight is at least ε/n . For every matching M , we have $w(M) - \varepsilon/2 \leq w'(M)$. As there exists one edge of weight 1, the effect of discretization of edge weights decreases the approximation factor by at most a factor of $(1 - \varepsilon/2)$.

Number of Distinct Gain-Indexes. The rounded edge weights are multiples of ε/n in the interval $[\varepsilon/n, 1]$. Let

$$w_{\min}(\varepsilon) \triangleq \min\{w(e) \mid w(e) \geq \varepsilon/n\}.$$

Note that $w_{\min}(\varepsilon) \geq \varepsilon/n$.⁹

As the edge weights are multiples of ε/n in the interval $[w_{\min}(\varepsilon), 1]$, it follows that the gains of $(M, [1, k])$ -augmenting paths are in the range $[w_{\min}(\varepsilon), k]$. Hence $(M, [1, k])$ -augmenting paths have at most $O(\log(k/w_{\min}(\varepsilon)))$ distinct gain-indexes.

8.2 CENTLOCAL-Implementation

CENTLOCAL-algorithm for index-greedy MIS. A sequential algorithm for computing an index-greedy MIS of G adds vertices to the MIS by scanning the vertices in nonincreasing gain-index order. We refer to this algorithm as IG-MIS. Following Section 4, a simulation of such a sequential algorithm is obtained by computing an acyclic orientation. For IG-MIS, the orientation is induced by the vertex coloring that is the Cartesian product of the gain-index of the vertex and its (regular) color. Lexicographic ordering is used to compare the colors. We summarize the probe complexity and probe radius of the CENTLOCAL-algorithm for IG-MIS in the following lemma (recall that ℓ denotes the number of distinct index-gains).

Lemma 23. *An index-greedy MIS can be computed by a CENTLOCAL-algorithm with the following properties:*

1. *The probe radius is $O(\Delta^2 \cdot \ell + \log^* n)$.*
2. *The probe complexity is $O(\Delta^{\Delta^2 \cdot \ell + 1} \cdot (\log^* n + \Delta^2)) = \Delta^{O(\Delta^2 \cdot \ell)} \cdot \log^* n$.*

Proof. The probe radius is simply the number of colors (in the Cartesian product) plus the radius of the regular Δ^2 -coloring algorithm. The number of colors is $\Delta^2 \cdot \ell$ and the radius of the Δ^2 -coloring algorithm is $O(\log^* n)$.

The probe complexity is bounded by the reachability of the orientation times the probe complexity of the regular Δ^2 -coloring algorithm. The reachability of the orientation is bounded by $\Delta^{\Delta^2 \cdot \ell}$. The probe complexity of the regular Δ^2 -coloring algorithm is $\Delta \cdot (\log^* n + \Delta^2)$, and the lemma follows. \square

⁹We remark that $w_{\min}(\varepsilon)$ may be much bigger than ε/n . For example, if w_{\min} is constant (say, $1/100$). The analysis of the probe complexity and the probe radius uses $1/w_{\min}(\varepsilon)$ instead of n/ε to emphasize the improved results whenever $1/w_{\min}(\varepsilon)$ is significantly smaller than $2n/\varepsilon$.

Note that the CENTLOCAL-algorithm computes a IG-MIS over $I(M_i)$ in which the class of a vertex equals its gain-index. As there are $O(\log(k/w_{\min}(\varepsilon)))$ distinct gain-indexes, it follows that we can apply Lemma 23 with $\ell = O(\log(k/w_{\min}(\varepsilon)))$ and $\Delta = \Delta(I(M_i))$.

CENTLOCAL implementation of the global algorithm. The implementation also uses the local improvement technique of Nguyen and Onak [NO08] repeating the same method used in Section 5. The pseudo-code of the CENTLOCAL-algorithm appears as Algorithms 8-10. This CENTLOCAL-algorithm implements Algorithm 7.

By induction, one can prove that $\mathcal{O}_i(e)$ computes membership of e in M_i . From Theorem 22 we obtain that \mathcal{O}_L is an $(1 - \varepsilon)$ -approximate CENTLOCAL-algorithm. The following theorem analyzes the probe complexity of \mathcal{O}_L (the theorem holds under the assumption that $w_{\min}(\varepsilon) < 1$).

Theorem 24. *There exists a CENTLOCAL $[\varphi]$ -algorithm for $(1 - \varepsilon)$ -approximate maximum weighted matching with*

$$\varphi = \left(\frac{1}{w_{\min}(\varepsilon)} \right)^{\Delta^{O(1/\varepsilon)}} \cdot (\log^* n)^{O(\frac{1}{\varepsilon} \cdot \log \frac{1}{\varepsilon})}$$

Proof. The analysis is similar to the one in Lemma 16. The key differences are as follows: (1) The augmenting paths in all recursive calls have length at most k . Hence the intersection graph is not the same graph in both algorithms. (2) A lexicographic-MIS is computed instead of an MIS. The analysis proceeds as follows.

$$\begin{aligned} |\mathcal{O}_i| &\leq |\mathcal{O}_{i-1}| + |\mathcal{A}_i| \\ &\leq |\mathcal{O}_{i-1}| + 2 \cdot \Delta^{2k+1} + \Delta^{2k+2} \cdot |\mathcal{O}_{i-1}| + |P_i(e)| \cdot |\text{IG-MIS}(I(M_{i-1}))| \cdot |\text{probe}(i)| \\ &\leq \Delta^{O(k)} \cdot |\mathcal{O}_{i-1}| + \Delta^{O(k)} \cdot (\Delta_i^{O(\Delta_i^2 \cdot \log(k/w_{\min}(\varepsilon)))} \cdot \log^* n_i) \cdot (\Delta^{O(k)} \cdot |\mathcal{O}_{i-1}|). \end{aligned}$$

Because $n_i \leq n^{2k+1}$ and $\Delta_i = O((2k+1)^2 \cdot \Delta^{2k+1})$, it follows that

$$\begin{aligned} |\mathcal{O}_L| &\leq \Delta^{\Delta^{O(k)} \cdot \log(1/w_{\min}(\varepsilon))} \cdot \log^* n \cdot |\mathcal{O}_{L-1}| \\ &= \Delta^{\Delta^{O(k)} \cdot \log(1/w_{\min}(\varepsilon))} \cdot (\log^* n)^L. \end{aligned} \tag{3}$$

Note that $\Delta^{\log(1/w_{\min}(\varepsilon))} = (1/w_{\min}(\varepsilon))^{\log(\Delta)}$, and the lemma follows. \square

9 A DISTLOCAL $(1 - \varepsilon)$ -Approximate MWM Algorithm

In this section, we present a DISTLOCAL-algorithm that computes a $(1 - \varepsilon)$ -approximate weighted matching. The algorithm is based on the same design methodology as in Section 6. Namely, we bound the probe radius of the CENTLOCAL-algorithm for MWM (see Lemma 26) and apply the simulation technique (see Proposition 1).

Theorem 25. *There is a deterministic DISTLOCAL $[r]$ -algorithm for computing a $(1 - \varepsilon)$ -approximate MWM with*

$$r_G(\mathcal{O}_L) \leq O\left(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\varepsilon} \cdot \log^* n\right) + \Delta^{O(1/\varepsilon)} \cdot \log\left(\frac{1}{w_{\min}(\varepsilon)}\right)$$

The proof of Theorem 25 is based on the following lemma. Recall that ignoring lightweight edges implies that $\frac{1}{w_{\min}(\varepsilon)} \leq \frac{n}{\varepsilon}$.

Lemma 26. *The probe radius of the CENTLOCAL-algorithm \mathcal{O}_L is*

$$r_G(\mathcal{O}_L) \leq O\left(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\varepsilon} \cdot \log^* n\right) + \Delta^{O(1/\varepsilon)} \cdot \log\left(\frac{1}{w_{\min}(\varepsilon)}\right)$$

Proof. The description of the oracle \mathcal{O}_i implies that the probe radius $r_G(\mathcal{O}_i)$ satisfies the following recurrence:

$$r_G(\mathcal{O}_i) = \begin{cases} 0 & \text{if } i = 0, \\ \max\{r_G(\mathcal{O}_{i-1}), r_G(\mathcal{A}_i)\} & \text{else.} \end{cases}$$

The description of the procedure \mathcal{A}_i implies that the probe radius $r_G(\mathcal{A}_i)$ satisfies the following recurrence:

$$r_G(\mathcal{A}_i) \leq O(k) + \max\{r_G(\mathcal{O}_{i-1}), r_G(\text{IG-MIS}(I(M_{i-1})))\}.$$

The probe radius of IG-MIS with respect to G satisfies

$$r_G(\text{IG-MIS}(I(M_{i-1}))) \leq O(k) \cdot r_{I(M_{i-1})}(\text{IG-MIS}(I(M_{i-1}))) + r_G(\text{probe}(i-1, p)).$$

By Lemma 23, $r_{I(M_{i-1})}(\text{IG-MIS}(I(M_{i-1}))) \leq O(\log^* n) + \Delta^{O(k)} \cdot \log(1/w_{\min}(\varepsilon))$.

The probe radius of a simulation of a probe to $I(M_{i-1})$ satisfies

$$r_G(\text{probe}(i-1, p)) \leq O(k) + r_G(\mathcal{O}_{i-1}).$$

It follows that

$$\begin{aligned} r_G(\mathcal{O}_i) &\leq r_G(\mathcal{O}_{i-1}) + O(k \cdot \log^* n) + \Delta^{O(k)} \cdot \log(1/w_{\min}(\varepsilon)) \\ &\leq i \cdot (O(k \cdot \log^* n) + \Delta^{O(k)} \cdot \log(1/w_{\min}(\varepsilon))), \end{aligned}$$

and the lemma follows. \square

10 Upper Bounds and Lower Bounds for O-RAD in the DISTLOCAL Model

In this section we consider DISTLOCAL-algorithms for computing orientations over bounded degree graphs. The goal is to find an orientation with the smallest possible radius (O-RAD). We first list DISTLOCAL $[\log^* n]$ -algorithms for O-RAD that are obtained from vertex coloring algorithms in which the radius of the orientation is polynomial in the maximum degree of the graph. We then prove that every orientation that computed in $o(\log^* n)$ rounds must have a radius that grows as a function of n . Thus, $\Theta(\log^* n)$ rounds are necessary and sufficient for computing an acyclic orientation with reachability that is bounded by a function of the maximum degree.

10.1 DISTLOCAL Algorithms for O-RAD

As observed in Proposition 2, every vertex coloring induces an acyclic orientation. This implies that a DISTLOCAL c -coloring algorithm can be used to compute an acyclic orientation with radius c by performing the same number of rounds. The distributed coloring algorithms [Lin92, Theorem 4.2] [BE09, Theorem 4.6] imply the following corollary.

Corollary 27. *There are DISTLOCAL algorithms for O-RAD with the following parameters:*

1. Radius $O(\Delta^2)$ in $O(\log^* n)$ rounds.
2. Radius $\Delta + 1$ in $O(\Delta) + \frac{1}{2} \log^* n$ rounds.

10.2 Lower Bound for O-RAD in the DISTLOCAL Model

In this section we consider the problem of computing an acyclic orientation H of a graph G with radius $\text{rad}(H)$ that does not depend on the number of vertices n (it may depend on Δ).

Definition 3. Let $g : \mathbb{N} \rightarrow \mathbb{N}$ denote a function. In the O-RAD(g)-problem, the input is a graph G with maximum degree Δ . The goal is to compute an orientation H of G with radius $\text{rad}(H) \leq g(\Delta)$ (if such an orientation exists).

Our goal is to prove the following theorem.

Theorem 28. *For every function g , there is no DISTLOCAL $[o(\log^* n)]$ -algorithm that solves the O-RAD(g)-problem.*

Proof. The proof is based on a reduction from MIS to O-RAD. Let G_n denote an undirected ring with n vertices. Let $g : \mathbb{N} \rightarrow \mathbb{N}$ be any function (e.g., Ackermann function). Assume, for the sake of contradiction, that there exists a DISTLOCAL $[r]$ -algorithm that computes an acyclic orientation H_n of G_n with radius $\text{rad}(H_n) \leq g(\Delta)$. Then, by Lemma 12 and Proposition 1 there is a DISTLOCAL $[r + g(\Delta)]$ -algorithm for MIS.

If $r = o(\log^* n)$, then this contradicts the theorem of Linial [Lin92] that states that there is no DISTLOCAL algorithm that computes an MIS over a ring in less than $\frac{1}{2} \cdot \log^* n$ rounds. \square

Remark 1. *Theorem 28 can be extended to $g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ that is a function of Δ and n . The dependency on n can be at most $o(\log^* n)$, while the dependency on Δ stays arbitrary.*

Remark 2. *Theorem 28 can be extended to randomized algorithms since the lower bound for MIS in [Lin92] holds also for randomized algorithms.*

11 Discussion

In this work we design centralized local algorithms for several graph problems. Our algorithms are deterministic, do not use any state-space, and the number of probes (queries to the graph) is $\text{poly}(\log^* n)$ where n is the number of graph vertices.¹⁰ Previously known algorithms for these problems make $\text{polylog}(n)$ probes, use $\text{polylog}(n)$ state-space, and have failure probability $1/\text{poly}(n)$. While a basic tool in previous works is (random) *vertex rankings*, our basic

¹⁰For approximate weighted matching, we require a constant ratio of maximum-to-minimum edge weight.

(seemingly weaker) tool, is *acyclic graph orientations with bounded reachability*. That is, our algorithms use as a subroutine a local procedure that orients the edges of the graph while ensuring an upper bound on the number of vertices reachable from any vertex. To obtain such orientations we employ a local *coloring algorithm* which uses techniques from local *distributed* algorithms for coloring.

On the other hand, by using a technique of Nguyen and Onak [NO08] that was introduced for local computation in the context of sublinear approximation algorithms, we get a new result in local distributed computing: A deterministic algorithm for approximating a maximum matching to within $(1 - \varepsilon)$ that performs $\Delta^{O(1/\varepsilon)} + O\left(\frac{1}{\varepsilon^2}\right) \cdot \log^* n$ rounds where Δ is the maximum degree in the graph. This is the best known algorithm for this problem for constant Δ . The technique also extends to approximate maximum weighted matching.

The probe complexity of any CENTLOCAL-algorithm A is bounded by $\Delta^{\text{rad}(A)}$, where $\text{rad}(A)$ denote the probe radius of A . Employing the above bound on the probe complexity of our CENTLOCAL-algorithms places the $\log^* n$ in the exponent. Our analyses of the probe complexity in the CENTLOCAL-algorithms is slightly stronger because it avoids having the $\log^* n$ in the exponent.

References

- [ARVX12] Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *SODA*, pages 1132–1139, 2012.
- [BE09] Leonid Barenboim and Michael Elkin. Distributed $(\Delta + 1)$ -coloring in linear (in Δ) time. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 111–120. ACM, 2009.
- [CHW08] Andrzej Czygrinow, Michal Hańćkowiak, and Wojciech Wawrzyniak. Fast distributed approximations in planar graphs. In *Distributed Computing*, pages 78–92. Springer, 2008.
- [CV86] Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986.
- [GPS88] Andrew V Goldberg, Serge A Plotkin, and Gregory E Shannon. Parallel symmetry-breaking in sparse graphs. *SIAM Journal on Discrete Mathematics*, 1(4):434–446, 1988.
- [HK73] John E Hopcroft and Richard M Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.
- [HV06] Stefan Hougardy and Doratha E. Vinkemeier. Approximating weighted matchings in parallel. *Inf. Process. Lett.*, 99(3):119–123, 2006.
- [Kuh09] Fabian Kuhn. Weak graph colorings: distributed algorithms and applications. In *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*, pages 138–144. ACM, 2009.
- [Lin92] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

- [LPSP08] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. In *SPAA*, pages 129–136, 2008.
- [LRY14] Reut Levi, Ronitt Rubinfeld, and Anak Yodpinyanee. Local computation algorithms for graphs of non-constant degrees. unpublished manuscript, 2014.
- [LW08] Christoph Lenzen and Roger Wattenhofer. Leveraging Linial’s locality limit. In *Distributed Computing*, pages 394–407. Springer, 2008.
- [MRVX12] Yishay Mansour, Aviad Rubinstein, Shai Vardi, and Ning Xie. Converting on-line algorithms to local computation algorithms. In *Automata, Languages, and Programming*, pages 653–664. Springer, 2012.
- [MV13] Yishay Mansour and Shai Vardi. A local computation approximation scheme to maximum matching. In *APPROX-RANDOM*, pages 260–273, 2013.
- [NO08] Huy N Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *Foundations of Computer Science, 2008. FOCS’08. IEEE 49th Annual IEEE Symposium on*, pages 327–336. IEEE, 2008.
- [ORRR12] Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *SODA*, pages 1123–1131, 2012.
- [Pel00] David Peleg. *Distributed computing: a locality-sensitive approach*, volume 5. SIAM, 2000.
- [PR01] Alessandro Panconesi and Romeo Rizzi. Some simple distributed algorithms for sparse networks. *Distributed computing*, 14(2):97–100, 2001.
- [PR07] Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1):183–196, 2007.
- [PS04] Seth Pettie and Peter Sanders. A simpler linear time $2/3$ -epsilon approximation for maximum weight matching. *Inf. Process. Lett.*, 91(6):271–276, 2004.
- [PS10] Alessandro Panconesi and Mauro Sozio. Fast primal-dual distributed algorithms for scheduling and matching problems. *Distributed Computing*, 22(4):269–283, 2010.
- [RTVX11] Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *ICS*, pages 223–238, 2011.
- [RV14] Omer Reingold and Shai Vardi. New techniques and tighter bounds for local computation algorithms. *CoRR*, abs/1404.5398, 2014.
- [Suo13] Jukka Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2):24:1–24:40, March 2013.
- [YYI12] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. Improved constant-time approximation algorithms for maximum matchings and other optimization problems. *SIAM J. Comput.*, 41(4):1074–1093, 2012.